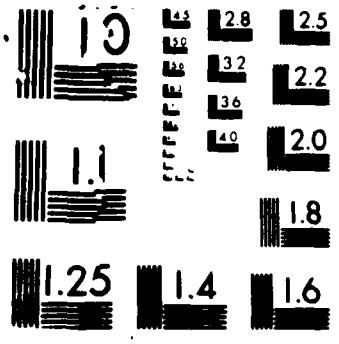
AD-A181 742 SOFTWARE DEVELOPMENT ENVIRONMENTS(U) CARNEGIE-MELLON 1/1
UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST
R J ELLISON JUL 86 CMU/SEI-86-TM-10 ESD-TR-86-217
UNCLASSIFIED F19628-85-C-0003 F/G 12/5 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

AD-A181 742

Technical Memorandum
SEI-86-TM-10

Carnegie-Mellon University
Software Engineering Institute

Software Development Environments

by
Robert Ellison

July 1986

SECURITY CLASSIFICATION OF THIS PAGE

AD-A181 972

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNLIMITED, UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT UNCLASSIFIED, UNLIMITED, DTIC, NTIS
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) SEI-86-TM-10	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-TR-86-217
6a. NAME OF PERFORMING ORGANIZATION SOFTWARE ENGINEERING INST. 6b. OFFICE SYMBOL (If applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI JOINT PROGRAM OFFICE
6c. ADDRESS (City, State and ZIP Code) CARENGIE-MELLON UNIVERSITY PITTSBURGH, PA 15213	7b. ADDRESS (City, State and ZIP Code) ESD/XRS1 HANSCOM AIR FORCE BASE HANSCOM, MA 01731
8a. NAME OF FUNDING/SPONSORING ORGANIZATION SEI JPO 8b. OFFICE SYMBOL (If applicable) ESD/XRS1	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628 85 0003
8c. ADDRESS (City, State and ZIP Code) CARNEGIE-MELLON UNIVERSITY PITTSBURGH, PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 63752F PROJECT NO.: N/A TASK NO.: N/A WORK UNIT NO.: N/A
11. TITLE (Include Security Classification) SOFTWARE DEVELOPMENT ENVIRONMENTS	
12. PERSONAL AUTHOR(S) ROBERT ELLISON	
13a. TYPE OF REPORT FINAL 13b. TIME COVERED FROM ... TO ...	14. DATE OF REPORT (Yr., Mo., Day) JULY 1986 15. PAGE COUNT 8
16. SUPPLEMENTARY NOTATION N/A	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

THE GROWING DEMAND FOR RELIABLE LARGE-SCALE SOFTWARE SYSTEMS CANNOT BE MET WITHOUT ADVANCES IN SOFTWARE DEVELOPMENT ENVIRONMENTS. ALTHOUGH PROMISING TECHNOLOGIES ARE EMERGING, A NUMBER OF ISSUES MUST BE ADDRESSED TO ENSURE THE TIMELY TRANSITION OF THOSE TECHNOLOGIES TO PRACTICE. THIS PAPER DISCUSSES ISSUES RELEVANT TO THE TRANSITION OF SUCH TECHNOLOGIES AND PROJECTS TO BE UNDERTAKEN BY THE SOFTWARE ENGINEERING INSTITUTE TO ADDRESS THOSE ISSUES.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☒ DTIC USERS ☒	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED, UNLIMITED, DTIC, NTIS
22a. NAME OF RESPONSIBLE INDIVIDUAL KARL H. SHINGLER	22b TELEPHONE NUMBER (Include Area Code) 412 268-7630 22c OFFICE SYMBOL SEI JPO

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

Technical Memorandum

SEI-86-TM-10
July 1986

Software Development Environments

by

Robert Ellison
Software Engineering Institute
Carnegie-Mellon University
Pittsburgh, PA 15213

Approved for Public Release. Distribution unlimited.

This work was sponsored by the Department of Defense.

Software Development Environments: Research to Practice

Robert J. Ellison

ABSTRACT The growing demand for reliable large-scale software systems cannot be met without advances in software development environments. Although promising technolgoies are emerging, a number of issues must be addressed to ensure the timely transition of those technologies to practice. This paper discusses issues relevant to the transition of such technologies and projects to be undertaken by the Software Engineering Institute to address those issues.

1 Introduction

During the last few years, we have seen the appearance of a number of software development environments for support of programming-in-the-large [3] [4]. The Software Engineering Institute at Carnegie-Mellon University has been tasked with speeding the transition of modern software engineering technology to practice [2]. Part of that effort has been devoted to examining large-scale software development environment efforts. This note represents the author's observations of that process as well as the comments expressed by the participants of the SEI workshop on the *Software Factory of the Future*, which was held in Morgantown, West Virginia in October, 1985. It also reflects the rationale that helps guide the definition of SEI projects on software development environments.

2 Transition of Environment Technology to Practice

While there has been significant research activity with respect to environment generation [10] [6], very little of that work has yet to appear in practice. Current large-scale environments are primarily hand crafted and often reflect the organization's internal software methodology. There are a wide variety of implementation schemes. Some organizations integrate a large collection of internally developed tools, while others have adopted a strategy of using commercial tooling. There is little commonality of the underlying system support for networks, databases, tooling, tool interface conventions, or user interfaces. The potential for sharing among these environments appears to be rather limited.

The SEI's primary mission is the transition of modern software engineering methods to practice. Software engineering environments represent a good means to support that end. They provide the means both to integrate tools and to provide a uniform conceptual framework for the user. While from one point of view an environment can enforce unform practices, it also provides the means to maintain the rich information base that most likely will be required to support reusability of requirements and designs. The SEI's role with respect to environments is not to build a specific environment, but to help explore the validity of new concepts by building prototypes, to stimulate the research community to attack critical problems, and to refine the requirements for the next generation of large-scale environments.

There are some pragmatic considerations that impact the design of industrial-grade environments that should be addressed if we want to see environments applied successfully to large-scale software development. The most critical issue may be the controlled evolution of the environment. Software development environments must adjust to new hardware such as workstations or graphical displays, and to software such as database technology or operating system support for distributing computing; they also must be able to reflect changes in the software process as the field of software engineering matures. In practice, an environment also must adjust over the life of a project to changes in project objectives or management policies. Within one organization, the demand will exist to tailor or extend the base-line environment to support needs specific to a project or application. This issue will be especially critical for an organization such as the SEI that has been charged with the task of speeding transition of research concepts to practice. While a specific environment might provide a good vehicle for the rapid introduction of Ada and associated software engineering practices, that same environment may be a significant impediment to the next wave of technological changes if it is not easy to modify or extend. Thus, while it may be hard to convince management to buy or build its first software development environment, it will probably be even harder to sell the second one given the initial investment. The state of the current practice and the relatively immature state of the theory for large-scale environments suggest that we would be naive to expect current environments to be long lived. On the other hand, it is not reasonable to continue to replace, rather than evolve, such expensive systems.

3 Scope: Environment Construction

A discussion of the construction of large-scale environments often turns to the management of scale. Such environments must not only manage a large amount of information, they also must coordinate the efforts of a large programming team. The issue is a natural one as many research prototypes have not demonstrated that they can manage a project even on the order of 100,000 lines of source code, and the existing industrial systems seem to stretch the capability of current database systems. Certainly scale will be one of the issues that the SEI will address, but several trends suggest that the scope of the environment should have higher priority in terms of increased functionality across the full life cycle as well as better integration between life-cycle phases.

Environment research in recent years has concentrated on the coding phase of software development and has been able to build on the theoretical foundations on language syntax and semantics. The design of full life-cycle environments does not yet have the equivalent underpinnings. The issue is not just the coverage of life-cycle phases, but the quality and type of tooling so that we can support the highly interactive and incremental style of processing demonstrated in [6], [9] or [10]. While work like [7] or [12] is a first step, there is nothing of equivalent maturity for dealing with the semantics of a full life-cycle environment. Certainly work in artificial intelligence or formal methods will apply, but it is precisely because an environment must mix such a variety of approaches that the design of the foundation becomes so critical.

The construction of environment generators has primarily been a research topic, but the prag-

matic considerations discussed above suggest that it will be critical to automate environment construction and that this topic should be given priority by the SEI. The size and complexity of modern environments demand resuable components. While first generation environment generators primarily concentrated on programming-in-the-small and the static semantics of ALGOL-class languages, they demonstrated the kind of reusability and extensibility that will be needed. Gandalf [6] attempted to address some of the issues involved with programming-in-the-large, and experience with that system [5] raised several issues that impact building large-scale environments.

The complexity of environments is driven by the desire to support the semantics of the software process in addition to the semantics of the implementation languages. A major goal of the Gandalf project was to address the semantic issues that arise with large-scale environments. The first environment prototypes constructed with the Gandalf system emphasized configuration management, version control, and project management, and the semantic issues raised by the effort is described in [7]. The Gandalf system is data driven. A procedure called an action routine is attached to each type of data and is responsible for maintaining consistency whenever that item is modified. Semantic information is effectively centralized in the action routine, and in most instances invisible to tool fragments. Whenever possible, tool activation is tied to the data rather than embedded in the tools themselves. This approach can make it easier to reuse tool fragments and to extend the system. This kind of separation will be more critical as we embed the semantics of the software process into environments. There are a variety of implementation schemes other than the Gandalf ones, such as a rule-based paradigm, that can obtain the same result. Kaiser [7] describes a more declarative approach.

Experience with environment generators, such as Gandalf, raises questions about the nature of tools in the next generation of environments. In most instances the user interface, the presentation of the data, and tool activation are separated from the tools themselves. That kind of architecture seems to be required if we are to manage effectively the change in environments as well as support tool fragments that are effectively reusable. That trend continues into new work such as Arcadia [12] and was one area of agreement at the Morgantown workshop. Such a tool architecture raises difficult problems in terms of moving such a concept to practice. If the style of tool construction is too different, then importation of existing tools will be difficult. This is an issue that organizations such as the SEI must address.

4 Scope: The Environment User

The kind of general paradigms for environments range from an intelligent software assistant to an automated software factory driven by formal methods. While the complexity of the design and implementation of a large-scale environment often gets the most attention, the issue of the complexity of the user interactions may be more critical and suggests that we should address more carefully the user side of large-scale environments. The issue is not so much human factors but the underlying cognitive models that best support the various tasks. The classic life cycle demonstrates the variety of tasks that must be addressed during software development. The

expectation is that the environment can provide integration across the full life cycle and better support users, particularly those in the maintenance or enhancement phase who must work across a variety of phases. While we seek common conventions with respect to the user interface across those tasks, it will be more important to have commonality with respect to the underlying cognitive models.

We should expect that the next generation of environments will support a variety of development paradigms. For example, the early phases of system design may be best attacked by an exploratory paradigm, while later development may be more closely controlled. In general, we should expect to see better support for incremental or evolutionary development and hence for tools that must work across life-cycle phases. Within a well-defined task such as design, there are a variety of paradigms. It may be appropriate to use both data-driven and process-driven methodologies for the same task. The limitations of our current technology or the shear effort of building an environment often lead us to support a single approach. The next generation of environments should support multiple paradigms.

5 SEI Projects

The SEI plans to address a number of the issues raised above. Environments are a continuing source of challenging research problems; but for a technology transition agent such as the SEI, the most difficult problem is to foster reasonable expectations for environments and to find a strategy for making productive use of a still maturing technology. It is not hard to draw the conclusion that the expectations are growing much more rapidly that our means to meet them. There appears to be an emerging consensus in the research community on the appropriate framework for the next generation of environments. Some pieces of that potential solution appear as prototypes, but certainly continued research is required for many topics.

One set of projects will address the infrastructure required of the advanced large-scale environments. The limited reusability that we now see among existing large-scale environments reflects the very limited commonality that we have with respect to user interfaces, database management systems, tool communication, and distributed computing that represent the infrastructure for environment construction. The SEI has set as one of its tasks the formation of a broader consensus on the infrastructure to make it easier to support reusability across environments and to establish a climate whereby the commercial marketplace can better contribute. The initial SEI efforts in this direction concentrate on support for distributed computing and on management of persistent data.

Distributed environments have been particularly difficult to design and build because of the limited support found in most operating systems. Production quality prototypes of distributed operating systems such as the Mach system at Carnegie-Mellon University [1] are now available; the increased functionally of such systems will impact both the implementation and functionality of environment tooling. The SEI has two functions here. On one hand, we need to make sure that the operating system requirements for large-scale environments are reflected in the next generation of operating systems. While heterogeneity of hardware and software may be a fact of life, it

will be important for a consensus, if not a standard, to be established for interprocess communication in distributed environments. The SEI will take an active role in that effort.

Management of persistent data will be an increasingly critical problem for environments that support extensive reusability or automation [8]. The level of technology represented by classical file systems and database management systems may not be sufficient to deal with these information management problems [8]. This is an area of on-going research and development where the SEI will be an active participant.

A second group of SEI projects will consider more domain-specific environments and concentrate more on user requirements. Real-time software is a major component of the Department of Defense software efforts. Such software usually must perform within severe constraints of efficiency, reliability, and resource utilization, and is generaly believed to be harder to specify and implement that other types of software. An environment for building real-time systems would appear to have high priority. On the other hand, there is no generally agreed on methodology for specifying, designing, implementing, and maintaining such software, and an environment construction project would be premature. The SEI effort in this area is twofold. One project will examine and evaluate existing methodologies for developing real-time software, perform a comparative critique of them, and recommend a methodology as the most suitable for use, both in general and specifically with Ada. A second project will examine some of the technical problems associated with real-time development environments such as support of debugging and testing.

Artificial intelligence has the potential for making major contributions to improving software development. AI can bring a different perspective to addressing complex problems, such as modeling intellectual processes, and contribute a fresh view on the software development process. It will be important for the SEI to follow those efforts that could impact the basic goals of environments. In addition, this project will address some of the issues we raised above with respect to providing support for multiple paradigms and to extending the notion of activity modeling to better reflect the semantics of the activity.

REFERENCES

[1] Mike Accetta, Robert Baron, William Bolosky, David Golub,Richard Rasid, Avadis Tevanian, and Michael Young.
Mach: A New Kernel Foundation for Unix Development.
In *Proceedings of USENIX Technical Conference*. Summer 1986.

[2] Mario R. Barbacci, A. Nico Habermann, Mary Shaw.
The Software Engineering Institute: Bridging Practice and Potential.
IEEE Software , November 1985.

[3] Barry W. Boehm, Maria H. Penedo, E. Don Stuckle, Robert D. Williams, and Arthur B. Pyster.
A Software Development Environment for Improved Productivity.
COMPUTER , June 1984.

[4] Frank DeRemer and Hans Kron.
Programming-in-the-Large Versus Programming-in-the-Small.
SIGPLAN Notices , June 1975.

[5] Robert. J. Ellison and Barbara J. Staudt.
The Evolution of the Gandalf System.
The Journal of Systems and Software , May 1985.

[6] A. N. Habermann and David S. Notkin.
The Gandalf Software Development Environment.
In *The Second Compendium of Gandalf Documentation.* Carnegie-Mellon University Computer Science Department, January 1982.

[7] Gail E. Kaiser.
Semantics for Structure Editing Environments.
PhD thesis, Carnegie-Mellon University Computer Science Department, 1985.

[8] John R. Nestor.
Toward a Persistent Object Base.
May, 1986.
This proceedings.

[9] Steven P. Reiss.
Graphical Program Development with PECAN Program Development Systems.
In *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.* ACM SIGSOFT/SIGPLAN, April 1984.

[10] Thomas Reps and Tim Teitelbaum.
The Synthesizer Generator.
In *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments.* April 1984.

[11] William E. Riddle and Lloyd G. Williams.
Software Environments Workshop Report.
SIGSOFT Software Engineering Notes , January 1986.

[12] Richard N. Taylor, Lori Clarke, Leon J. Osterweil, Jack C. Wileden, Alex Wolf and Michal Young.
Arcadia: A Software Development Environment Research Project.
January 1986.
Carnegie-Mellon University.

END

7-87

DTIC